

Rules of Acquisition for Mementos and Their Content

Shawn M. Jones
Los Alamos National Laboratory
Los Alamos, NM, USA
smjones@lanl.gov

Harihar Shankar
Los Alamos National Laboratory
Los Alamos, NM, USA
harihar@lanl.gov

ABSTRACT

Text extraction from web pages has many applications, including web crawling optimization and document clustering. Though much has been written about the acquisition of content from live web pages, content acquisition of archived web pages, known as mementos, remains a relatively new enterprise. In the course of conducting a study with almost 700,000 web pages, we encountered issues acquiring mementos and extracting text from them. The acquisition of memento content via HTTP is expected to be a relatively painless exercise, but we have found cases to the contrary. We also find that the parsing of HTML, already known to be problematic, can be more complex when one attempts to extract the text of mementos across many web archives, due to issues involving different memento presentation behaviors, as well as the age of the HTML in their mementos. For the benefit of others acquiring mementos across many web archives, we document those experiences here.

Categories and Subject Descriptors

H.3.7 [Digital Libraries]: Web Archives, Memento—*Systems Issues*

Keywords

Digital Preservation, HTTP, Resource Versioning, Web Archiving, HTML

1. INTRODUCTION

Text extraction of web page content is often done for web crawling optimization [14, 9] and document clustering [11, 15]. This paper documents the experience of comparing the text of archived web pages, hereafter referred to as **mementos**. We believe that this is an unprecedented attempt at comparing mementos from almost 700,000 web pages from across much of the lifespan of the web (1997-2012) and offers insights into the issues of using the content of different web archives for research purposes.

Using the same data as [13], we extracted more than 1 million URIs. Because not all pages were archived sufficiently for our experiments (i.e. not enough mementos), we were left with 680,136 references. Mementos from these remaining references come from more than 20 web archives.

Other studies in web page text extraction focused on comparing the text from downloaded live pages. Studies exist that focus on extracting text from mementos in a single web

archive [1, 10], but we are attempting to acquire and extract text from mementos across multiple web archives. As a side-effect, we have discovered that each web archive modifies the content of mementos for the purposes of branding or user experience. In terms of HTML, some mementos for the same live page are not presented consistently across different web archives. These differences in the behavior of web archives with respect to the presentation of mementos and the challenges involved are the focus of this paper.

2. COMPARING MEMENTOS

When acquiring memento content, the following steps must be completed:

- **Acquisition of Web Resource**, where the URI of the web resource is dereferenced and the content then downloaded for use.
- **Text Extraction**, where the tags (for HTML) and control characters (for PDFs) are removed from the content, producing only text for review.

We start with instances where downloading the content, given a URI, turns out to be more complex than anticipated.

2.1 Acquisition of Web Resource

Most mementos were easily acquired using cURL¹ and standard HTTP against a given web archive. Upon examination of the content of mementos acquired from some archives, however, it became clear that cases exist where an HTTP GET alone is not sufficient to acquire the content of a memento. For WebCite, none of their mementos were directly accessible by using HTTP GET alone. In addition, the Internet Archive replaces HTTP redirects with special pages that use JavaScript to simulate the redirect in a browser. Also, we encountered pages where the redirection was embedded within the page content itself, noted here as *HTML Meta Tag Refresh*.

2.1.1 Acquiring Mementos From WebCite

WebCite² was created to address the link rot of web at large references and contains quite a few mementos for the URIs in our collection. The first issue we encountered with WebCite was reliability. As seen in the web browser screenshot

¹<http://curl.haxx.se>

²<http://www.webcitation.org>

Listing 1: PhantomJS Script to Acquire Memento Content From WebCite

```
var page = require("webpage").create();
var system = require("system");

var wc_url = system.args[1];

page.onError = function(msg, trace) {
};

var response = [];

page.onResourceReceived = function(res) {
    response.push(res);
};

page.open(wc_url, function() {
    page.switchToFrame("main");
    console.log(page.frameContent);
    var headers = "";
    for (var i=0, r; r=response[i]; i++) {
        headers = "HTTP/1.1 " + r.status + "\n";
        if (r.url == "http://www.webcitation.org/mainframe.php") {
            for (var l=0, k; k=r.headers[l]; l++) {
                headers += k.name + ": " + k.value + "\n";
            }
            break;
        }
    }
    console.log("\n$$$$$$\n");
    console.log(headers);
    phantom.exit();
});
```

Figure 1: Example of a Failure In Dereferencing a Memento From WebCite

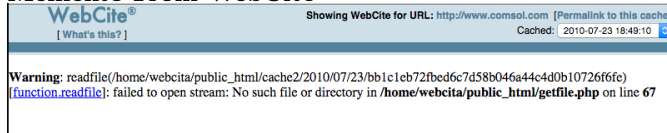
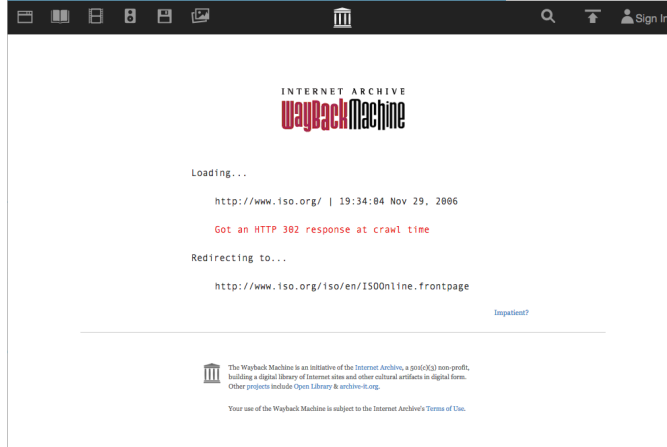


Figure 2: Example of a JavaScript Redirect from the Internet Archive



of Figure 1, visiting URIs on this site produces temporary errors.

Aside from these temporary errors, we observed that merely using cURL to dereference a WebCite URI does not result in the expected memento content. Dereferencing two different WebCite URIs result in the exact same HTML content, as seen in Appendix A. Based on our observations, the resulting HTML frameset loads the `topframe.php` and `mainframe.php` resources, then `mainframe.php` accepts a cookie containing session information to determine which content to load before redirecting the user to a page just containing the content in `mainframe.php`.

We turned to PhantomJS³ to help solve this problem; the script in Listing 1 handles this issue. After developing this script, we then perceived rate limiting from WebCite. We could find no documentation about rate limits, and requests for information from the WebCite webmaster were met with no response. We overcame the rate limits by running several copies of the PhantomJS script from different networks and consolidating the results. Due to the issues of using PhantomJS, which is slower than merely using cURL with HTTP GET, as well as the rate limiting, acquiring almost 100,000 mementos from WebCite took more than 1 month to complete. For comparison, more than 1 million mementos were acquired from the Internet Archive in 2 weeks.

2.1.2 Unresolved JavaScript Redirects

Most mementos from the Internet Archive fully support the expected HTTP status codes and the content from the majority of mementos was captured without incident. Approx-

³<http://phantomjs.org>

imately 84,000 consisted of pages that resembled Figure 2. Rather than merely issuing an HTTP 302 status code to redirect the web client, the Internet Archive has constructed these web pages that simulate a redirect in a web browser after some delay.

Once we detected these redirects, we used string matching to extract the redirect URIs from each page and then requested the content using that URI instead. Unfortunately, some of these pages were a redirect to a redirect, and so content for almost 8,000 could not be acquired. To solve this issue, one could create a PhantomJS client to download the target content of the chain of redirects. Seeing as less than 1% of mementos had additional JavaScript redirects, we excluded them from our dataset rather than creating this custom client.

2.1.3 HTML Meta Tag Refresh

HTML includes the `meta`⁴ element for representing metadata about a web page. It provides the `http-equiv` attribute⁵, allowing the web page author to either reload the current page after a certain number of seconds, or load a different page in place of the current page. The use of the `http-equiv` attribute has been deprecated for this purpose⁶, and is considered a bad practice for redirection because its use interferes with the history capability of some browsers⁷. Seeing as our mementos come from most of the history of the web, some exist from a time when this guidance was not given or well known.

In order to resolve this issue, one would need to load each page in a browser, or simulated browser, such as PhantomJS, and process the HTML on the page before being redirected. Seeing as less than 1% of mementos utilized this tag, we excluded them from our dataset rather than creating this custom client.

2.2 Text Extraction

Acquiring memento content from web archives was difficult in some cases, but the resulting mementos contained a large variety of issues, some caused by the age of the pages and others by the web archives themselves.

We are also aware of the capability provided by Wayback installations, that allows access to the original archived content by inserting the `_id` into a memento URI. Not all archives use Wayback, so the issues encountered here still need to be addressed by anyone engaging in text extraction across archives.

To acquire the content from the mementos, we attempted to use the Python screen-scraping library BeautifulSoup⁸, but found it was unsuccessful in extracting text from some pages, so we instead relied upon the HTML module of the

⁴<https://www.w3.org/TR/html-markup/meta>

⁵<https://www.w3.org/TR/html-markup/meta.http-equiv.refresh.html#meta.http-equiv.refresh>

⁶<https://www.w3.org/TR/WCAG10-HTML-TECHS/#meta-element>

⁷<https://www.w3.org/QA/Tips/reback>

⁸<http://www.crummy.com/software/BeautifulSoup/>

Listing 2: BeautifulSoup Code For Removing JavaScript and Stylesheets From Web Pages That Does Not Work In All Cases

```
soup = BeautifulSoup(content, "lxml")

# rip out JavaScript and CSS
for s in soup(["script", "style"]):
    s.extract()

# extract text
textonly = soup.get_text()
```

stricter lxml⁹ library. For PDF documents, we used the Python library PyPDF¹⁰.

Extracting only text was chosen for the following reasons:

1. We were aware that some web archives include additional HTML tags for branding and usability. If two mementos come from different archives, then these additional tags and content could alter the results if compared.
2. When comparing mementos, not all similarity measures function the same way. Some function at the byte level, while others rely upon more semantic constructs, like words and n-grams. Some do not process tags, as seen in HTML, or control characters as found in PDFs.
3. One could extract the contents of PDFs and HTML documents for comparison or topic analysis.

2.2.1 Replacing BeautifulSoup By lxml

We were aware that web archives included additional JavaScript and CSS in memento content for branding and usability purposes. We originally attempted to use the BeautifulSoup code shown in Listing 2 to remove these items, but found that it did not work in all cases, often resulting in output containing no content. An issue report was submitted to the BeautifulSoup maintainers in response to this issue¹¹. Instead, we used the HTML module of the more strict lxml library, shown in Listing 3, which was tested by visually inspecting the output of approximately 200 mementos.

2.2.2 Removing Archive-Specific Additions

Once we had confirmed the more reliable behavior of the lxml library, we could then use it to remove items added by specific archives. Using visual inspection and browser tools, we were able to identify specific blocks of HTML code that could be easily stripped from mementos without removing the actual content of the page.

Much of the archive-specific content can be removed by eliminating entire div elements, selected by their id identifier attribute, from the resulting memento.

⁹<http://lxml.de/lxmlhtml.html>

¹⁰<http://pybrary.net/pyPdf/>

¹¹<https://bugs.launchpad.net/beautifulsoup/+bug/1489208>

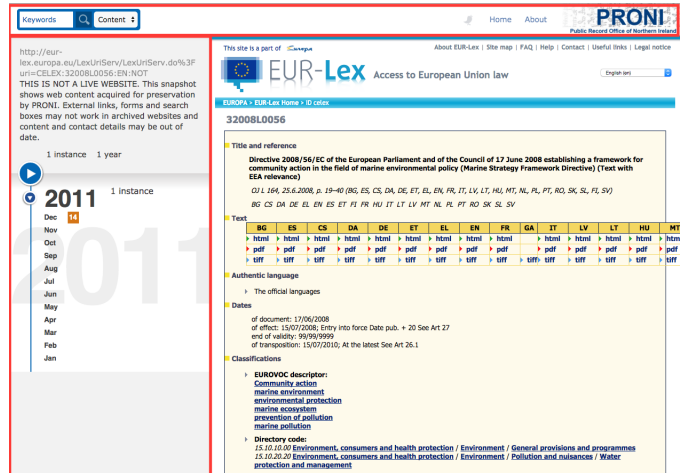
Figure 3: Example of the Wayback Toolbar from http://web.archive.org/web/20081126132802/http://www.bnl.gov/bnlweb/pubaf/pr/PR_display.asp?prID=05-38



Figure 4: Example of the National Archives Banner from <http://webarchive.nationalarchives.gov.uk/20120405114247/http://www.decc.gov.uk/en/content/cms/statistics/publications/flow/flow.aspx>



Figure 5: Example of a memento from the PRONI archive, with archive-specific elements outlined in red, URI shown is <http://webarchive.proni.gov.uk/20111214024729/http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?%3Furi=CELEX:32008L0056:EN:NOT>



All Wayback¹² sites (e.g. the Internet Archive¹³) add their banner, shown in Figure 3, inside the division with the `wm-ipp` identifier.

The UK National Archives adds a banner, shown in Figure 4, inside the `div` identified by `webArchiveInfobox`. In addition, the string `[ARCHIVED CONTENT]`, appearing at the beginning after text extraction, must be stripped out of the resulting text.

The PRONI web archive adds a banner and a sidebar, outlined in red in Figure 5. All of this added content is inside a division named `PRONIBANNER`. The `[ARCHIVED CONTENT]` string also must be removed from the beginning of the extracted text.

Archive.is, however is somewhat unique in its approach. Figure 6 shows a memento from the Archive.is archive, which places a header and sidebar on each memento, wrapping the memento content entirely and providing the reader with the ability to quickly scroll through a page by using the

¹²<http://www.netpreserve.org/openwayback>

¹³<http://archive.org/web/>

Listing 3: lxml Code for Removing JavaScript and Stylesheets From Web Pages

```
fs = lxml.html.document_fromstring(content)

# rip out JavaScript
for element in fs.iter("script"):
    element.drop_tree()

# rip out CSS
for element in fs.iter("style"):
    element.drop_tree()

... removal of archive-specific tags here ...

# extract text
textonly = fs.text_content()
```

Listing 4: Python Code for Removing Archive-Specific Elements From Web Pages

```
# National Archives-specific stuff
for element in fs.iter("div"):
    if element.get("id") == "webArchiveInfobox":
        element.drop_tree()

# Wayback-specific stuff
for element in fs.iter("div"):
    if element.get("id") == "wm-ipp":
        element.drop_tree()

# PRONI-specific stuff
for element in fs.iter("div"):
    if element.get("id") == "PRONIBANNER":
        element.drop_tree()

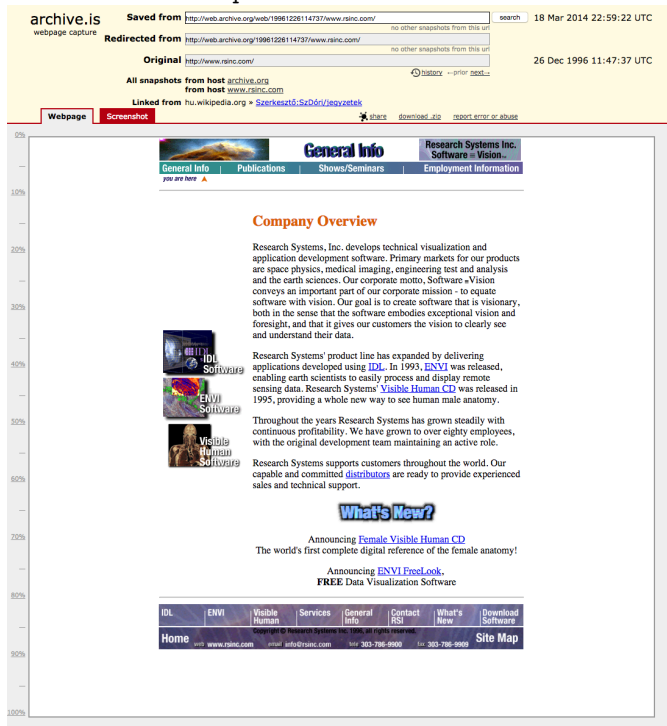
# Archive.is
if '<meta property="og:site_name" content="archive.is"' in content:
    for element in fs.iter("div"):
        if element.get("id") == "HEADER":
            element.drop_tree()

    for element in fs.iter("table"):
        if element.get("id") == "hashtags":
            element.drop_tree()

textonly = fs.text_content()

# PRONI adds [ARCHIVED CONTENT]
# National Archives adds [ARCHIVED CONTENT]
if "PRONIBANNER" in content or "webarchiveInfobox" in content:
    if textonly[0:19] == "[ARCHIVED CONTENT] ":
        textonly = textonly[19:]
```

Figure 6: Example of a memento from the Archive.is archive, URI shown is <http://archive.is/19961226114737/http://www.rsinc.com/>



links on the right. The bar and header are easily removed by eliminating the `div` elements with the identifiers **HEADER** and **hashtags**. Seeing as the original web page content of the memento may use the **HEADER** identifier, the code shown in Listing 4 only removes these elements if there is a corresponding meta tag identifying the memento as coming from Archive.is.

Appendix B contains example HTML snippets showing each of these sections for these archives. In the case of other archives, they are either already covered by these cases, or the removal of all JavaScript and CSS eliminates any additional archive-specific content.

2.2.3 Character Encoding Problems

In order to properly compare the text of documents, they must be rendered properly so that characters can be consistently compared. HTTP provides the **Content-Type** header [7] to allow web clients to interpret the received content with the appropriate parsing and rendering software. In order to compare documents, we relied upon the **Content-Type** header to indicate which library, lxml for `text/html` or pyPDF for `application/pdf`, is used to extract text from the given document. For the content-type `text/html`, there is an additional `charset` attribute that helps a client render HTML in the correct character encoding. Many character encodings exist¹⁴, allowing the browser to render different languages and other symbols. For HTML documents, we additionally used the `charset` value to decode the content

¹⁴<http://www.iana.org/assignments/character-sets/character-sets.xhtml>

of a given resource so that it would be rendered correctly for comparison. Decoding is necessary for document comparison and topic analysis.

Even though many character sets exist, UTF-8 [6] is currently the most widely used¹⁵. Even though the world has largely moved to UTF-8 for rendering all symbols, our dataset consists of web pages that exist before UTF-8 was adopted. We observed that, for some of our mementos, web archives did not respond with the correct `charset` value, resulting in decoding errors.

To address this issue, we used the following algorithm to detect the character encoding¹⁶ for an HTML document.

1. Attempt to extract the character set has been specified by the **Content-Type** HTTP header
 - (a) If a character set can be extracted, save it for use
 - (b) If no character set supplied, save character set as UTF-8 because UTF-8 is conceptually treated as a superset of several other character encodings¹⁷
2. Acquire the contents of the file using the saved character set
 - (a) If the file is XHTML¹⁸, attempt to extract the encoding from the XML encoding declaration and save it for use
 - (b) Try to decode the file contents using the saved character set
 - (c) If the saved character set fails, save character set as UTF-8 for the same superset reason
 - (d) If even UTF-8 does not work, throw an error

Using this algorithm we were able to effectively evaluate most of the mementos. Less than 500 still had decoding errors, but were discarded because they were less than 1% of the whole. It is possible that one could use confidence-based tools¹⁹, such as the `chardet`²⁰ library to guess the character encoding for these few that are left.

2.2.4 Whitespace Inequality

We learned that not all archives faithfully reproduce the archived content as it was originally captured. In addition to adding additional tags and content for branding and usability, archives also do not necessarily preserve the spacing in an HTML document. For example, Archive.is minifies²¹ all HTML, removing any additional space or newlines added by the author for legibility. Because Archive.is removes this whitespace, the entire HTML document is presented as a

¹⁵<https://googleblog.blogspot.com/2008/05/moving-to-unicode-51.html>

¹⁶https://en.wikipedia.org/wiki/Charset_detection

¹⁷<http://www.unicode.org/standard/principles.html>

¹⁸<http://www.w3.org/TR/xhtml1/>

¹⁹<http://userguide.icu-project.org/conversion/detection>

²⁰<https://pypi.python.org/pypi/chardet>

²¹[https://en.wikipedia.org/wiki/Minification_\(programming\)](https://en.wikipedia.org/wiki/Minification_(programming))

single line, making direct content comparison with another archive impossible. In contrast, the Internet Archive appears to reproduce the archived content very close to its original form, even including control characters added by HTML editors.

After text extraction, we replaced all whitespace with newlines for consistency in comparison between archives.

2.2.5 Null characters in HTML

In attempting to extract text from mementos, we expected PDFs to contain control characters and noted that the pyPDF library processes them correctly. We did not expect HTML to contain control characters, especially null characters. In some cases, if these characters had not been removed, then individual tags could not be parsed by the lxml library. In one example, a tag such as `<html>` was actually present in the file as `<[00]h[00]t[00]m[00]l[00]>`, where `[00]` represents a null character. Because null characters are still characters, the parsing library did not detect a valid HTML tag if the null characters appeared inside one.

This issue was resolved by removing all null characters from HTML files prior to parsing them with lxml.

2.2.6 Noscript With HTML-Encoded-HTML

The `noscript` element²² is used to provide alternative content for user agents that do not support scripting. Typically they present no issues to parsers, but in trying to process several mementos, we found that the lxml library could not extract text from 5,674 references due to a problematic combination of HTML elements with the `noscript` tag.

HTML contains elements consisting of start tags, but no end tags. The standard establishes a finite set of void elements²³ (e.g. `img`, `hr`). The `p` and `li` tags are also a special case whereby the end tag is optional²⁴ ²⁵. Even the `body` end tag can be omitted in some cases²⁶. Due to the optional nature of end tags for certain HTML cases, it appears that the lxml library accepts the `html` tag as the end of the document, even if some tags are not closed, and can arrive in a state where one can remove all content from a page by choosing the wrong element to remove.

This results in problems removing tags and extracting text, as shown in Listing 5. Telling lxml to remove all `noscript` elements results in the removal of everything from line 7 to the end of the page because the parser ends up in a state where it cannot find the end of this first `noscript` tag. One could just remove all tags in this example and extract the text, but if comparing the text with other mementos, we have found that some web archives do not present the contents of `noscript` elements the same way. As seen in Listing 6, some web archives present a memento for the same URI with the `<` and `>` replaced by `<` and `>`. Because `<` and `>` are interpreted as text and not the opening and closing of

a tag, text extraction returns these “faux tags”, thus skewing any comparison one intends to do between mementos coming from archives that exhibit inconsistent presentation behavior.

Considering we encountered this scenario in less than 1% of references, we discarded these items. We did not find a parser that could handle this issue.

3. CONCLUSIONS

We were surprised to discover these issues during the acquisition and extraction of text from mementos. Seeing as so many others have had success in comparing web pages, we assumed that mementos would be no different.

Instead, we found that acquisition of mementos was problematic in approximately 25% of our data set. We dispelled the assumption that all web archives honor the HTTP protocol in full. We observed that the Internet Archive, instead of preserving the original HTTP 302 status encountered when archiving a web page and creating a memento, generates a specialized HTML page with a JavaScript redirect that can only be followed once the content of the resulting specialized page is downloaded, processed, and executed. We exposed that WebCite, an on-demand archive for preserving references, does not even return the memento content directly to the user, resulting in the need to again download, process, and execute content from a web page in order to acquire the content of a memento. This behavior is not RESTful [8] because it requires content inspection and additional processing outside of HTTP to acquire a resource. The requirement to utilize a web browser, rather than HTTP only, for the acquisition of web content is common for live web content, as detailed by Kelly [12] and Brunelle [3], but we did not anticipate that we would need a browser simulation tool, such as PhantomJS, to acquire memento content.

The acquisition of mementos is key to the success of many efforts, such as Memento [16] and Hiberlink [4], that seek to combat link rot and content drift. Most readers of web content are trying to access web archives for a better understanding of the context of the time period in which these resources were archived. Problems accessing mementos by use of HTTP indicates the possibility of potential future issues with web browsers, preventing these readers from achieving their goal.

When processing the content of a memento, we exposed several issues. Some mementos could not be processed by the popular HTML processing library Beautiful Soup, leading us to use lxml instead. We documented how to remove archive-specific content for the archives in our experiment. We found that character encoding can be inaccurate for mementos, resulting in the creation of a special algorithm for acquiring the character encoding, falling back to UTF-8 if that failed. We observed that different web archives present mementos to the user with different levels of faithful reproduction, resulting in differences in whitespace. We encountered mementos containing null characters that made parsing the HTML tags difficult. Finally, we encountered mementos that could not be parsed by lxml at all because the DOM tree was corrupted by the use of `noscript` elements and unclosed tags.

²²<https://www.w3.org/TR/html-markup/noscript.html>

²³<https://www.w3.org/TR/html-markup/syntax.html#void-element>

²⁴<https://www.w3.org/TR/html-markup/p.html>

²⁵<https://www.w3.org/TR/html-markup/li>

²⁶<https://www.w3.org/TR/html-markup/body>

Listing 5: Simplified `<noscript>` Problem Example

```
1 <html>
2 ...
3 <body>
4 ...
5
6
7 <noscript> ← lxml parser encounters this open tag
8     <table border="0">
9 </noscript>
10
11 ... Much of the content for this page goes here ...
12
13 <noscript>
14     </table>
15 </noscript> ← lxml considers this to be the closing tag of the second noscript
16
17 </body>
18 </html> ← lxml assumes that we will never get to the closing tag of the first noscript
```

Listing 6: Simplified `<noscript>` Example, With `<` and `>` as Seen in Mementos from Some Archives

```
1 <html>
2 ...
3 <body>
4 ...
5
6
7 <noscript> ← lxml parser encounters this open tag
8     &lt;table border="0"&gt;
9 </noscript>
10
11 ... Much of the content for this page goes here ...
12
13 <noscript>
14     &lt;/table&gt;
15 </noscript> ← lxml considers this to be the closing tag because of the open table tag in
16     between
17 </body>
18 </html>
```


It is obvious that some issues are a result of poor source material (e.g. the original page had improperly formatted HTML), but others, such as spacing, are introduced by the archives themselves. Studies have been performed using text extraction on web pages, including mementos, but the effectiveness of text extraction is also quite important to the development of efforts for searching [2] and semantic analysis [5] as well as archiving. Without reliable support for text extraction, many of these efforts will be incomplete at best and fail at worse.

We have presented these issues to raise awareness so that future projects may benefit from these experiences and solutions.

4. REFERENCES

- [1] Y. AlNoamany, M. Weigle, and M. Nelson. Detecting off-topic pages in web archives. In S. Kapidakis, C. Mazurek, and M. Werla, editors, *Research and Advanced Technology for Digital Libraries*, volume 9316 of *Lecture Notes in Computer Science*, pages 225–237. Springer International Publishing, 2015.
- [2] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International Conference on World Wide Web 7, WWW7*, pages 107–117, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.
- [3] J. F. Brunelle, M. Kelly, M. C. Weigle, and M. L. Nelson. The impact of javascript on archivability. *International Journal on Digital Libraries*, pages 1–23, 2015.
- [4] P. Brunhill, M. Mewissen, T. Strickland, R. Wincewicz, C. Grover, B. Alex, R. Tobin, C. Metheson, and H. Van de Sompel. hiberlink. <http://hiberlink.org>, July 2015.
- [5] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134, 2005.
- [6] F. Yergeau. UTF-8, a transformation format of ISO 10646, Internet RFC 3629. <http://tools.ietf.org/html/rfc3629>, November 2003.
- [7] R. Fielding and J. Reschke. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, Internet RFC 7231. <http://tools.ietf.org/html/rfc7231>, June.
- [8] R. T. Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.
- [9] C. Figuerola, R. Díaz, J. Alonso Berrocal, and A. Zazo Rodríguez. Web document duplicate detection using fuzzy hashing. In J. Corchado, J. Pérez, K. Hallenborg, P. Golinska, and R. Corchuelo, editors, *Trends in Practical Applications of Agents and Multiagent Systems*, volume 90 of *Advances in Intelligent and Soft Computing*, pages 117–125. Springer Berlin Heidelberg, 2011.
- [10] A. N. Jackson. Ten years of the UK web archive: what have we saved? <http://anjackson.net/2015/04/27/what-have-we-saved-iipc-ga-2015/>, Apr 2015.
- [11] A. Kavitha Karun, P. Mintu, and K. Lubna. Comparative analysis of similarity measures in document clustering. In *Green Computing, Communication and Conservation of Energy (ICGCE), 2013 International Conference on*, pages 857–860, Dec 2013.
- [12] M. Kelly, J. F. Brunelle, M. C. Weigle, and M. L. Nelson. *Research and Advanced Technology for Digital Libraries: International Conference on Theory and Practice of Digital Libraries, TPDL 2013, Valletta, Malta, September 22-26, 2013. Proceedings*, chapter On the Change in Archivability of Websites Over Time, pages 35–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [13] M. Klein, H. Van de Sompel, R. Sanderson, H. Shankar, L. Balakireva, K. Zhou, and R. Tobin. Scholarly context not found: One in five articles suffers from reference rot. *PLoS ONE*, 9(12):e115253, 12 2014.
- [14] G. S. Manku, A. Jain, and A. Das Sarma. Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web, WWW '07*, pages 141–150, New York, NY, USA, 2007. ACM.
- [15] N. Sandhya and A. Govardhan. Analysis of similarity measures with wordnet based text document clustering. In S. Satapathy, P. Avadhani, and A. Abraham, editors, *Proceedings of the International Conference on Information Systems Design and Intelligent Applications 2012 (INDIA 2012) held in Visakhapatnam, India, January 2012*, volume 132 of *Advances in Intelligent and Soft Computing*, pages 703–714. Springer Berlin Heidelberg, 2012.
- [16] H. Van de Sompel, M. L. Nelson, and R. Sanderson. HTTP Framework for Time-Based Access to Resource States – Memento, Internet RFC 7089. <https://tools.ietf.org/html/rfc7089>, December 2013.

APPENDIX

A. WEBCITE EXAMPLES

Listing 7: HTML From WebCite URI <http://www.webcitation.org/6BToD7Sud>

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-frameset.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>WebCite query result</title>
    <link rel="stylesheet" type="text/css" href="/basic.css" />
      <link rel="stylesheet" type="text/css" href="/nicetitle.css" />
    <script src="https://www.google.com/recaptcha/api.js" async defer</script>
  </head>
    <frameset rows="60,*" frameborder="0">
      <frame src="/topframe.php" name="nav" noresize="
        noresize" marginwidth="0" marginheight="0"
        scrolling="no" />
      <frame src="/mainframe.php" name="main" noresize=
        "noresize" marginwidth="0" marginheight="0" />
    </frameset>
  </html>
```

Listing 8: HTML From WebCite URI <http://www.webcitation.org/5rRjz19dY>, a different URI, but with output identical to Listing 7

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-frameset.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>WebCite query result</title>
    <link rel="stylesheet" type="text/css" href="/basic.css" />
      <link rel="stylesheet" type="text/css" href="/nicetitle.css" />
    <script src="https://www.google.com/recaptcha/api.js" async defer</script>
  </head>
    <frameset rows="60,*" frameborder="0">
      <frame src="/topframe.php" name="nav" noresize="
        noresize" marginwidth="0" marginheight="0"
        scrolling="no" />
      <frame src="/mainframe.php" name="main" noresize=
        "noresize" marginwidth="0" marginheight="0" />
    </frameset>
  </html>
```

B. HTML SNIPPET EXAMPLES OF ARCHIVE-SPECIFIC CONTENT

Listing 9: HTML Snippet from Internet Archive URI http://web.archive.org/web/20081126132802/http://www.bnl.gov/bnlweb/pubaf/pr/PR_display.asp?prID=05-38, showing the beginning of the wm-ipp section

```
...
<div id="wm-ipp" lang="en" style="display:none;">

<div style="position:fixed;left:0;top:0;width:100%!important">
<div id="wm-ipp-inside">
  <table style="width:100%;"><tbody><tr>
    <td id="wm-logo">
      <a href="/web/" title="Wayback Machine home page"></a>
    </td>
    <td class="c">
      <table style="margin:0 auto;"><tbody><tr>
        <td class="u" colspan="2">
          <form target="_top" method="get" action="/web/form-submit.jsp" name="wmtb" id="wmtb
"><input type="text" name="url" id="wmtbURL" value="http://www.bnl.gov/bnlweb/
pubaf/pr/PR_display.asp?prID=05-38" style="width:400px;" onfocus="this.focus();
this.select();" /><input type="hidden" name="type" value="replay" /><input type=
"hidden" name="date" value="20081126132802" /><input type="submit" value="Go" />
          <span id="wm_tb_options" style="display:block;"></span></form>
        </td>
        <td class="n" rowspan="2">
          <table><tbody>
            <!-- NEXT/PREV MONTH NAV AND MONTH INDICATOR -->
            <tr class="m">
              <td class="b" nowrap="nowrap">
                ...
```

Listing 10: HTML Snippet from UK National Archives URI <http://webarchive.nationalarchives.gov.uk/20120405114247/http://www.decc.gov.uk/en/content/cms/statistics/publications/flow/flow.aspx>, showing the beginning of the webArchiveInfobox section

```
...
<div id="webArchiveInfobox" style="display: block; min-width: 130%; margin-left:-100px;
height: 110px; position: absolute; top: -110px; left: -100px; width:1400px; padding: 0
px; margin: 0px; background-color: #CC0000; opacity: 1; overflow: hidden;">
<script type="text/javascript">
  BANNER.TNA_VERSION="13/5/2014";
  function enforceBanner() {
    thebody = document.getElementsByTagName("body")[0];
    if (thebody != null && thebody.style != null) {
      //inject style to override body with id
      thebody.style.backgroundColor = "0px 73px";
      thebody.style.position = "relative";
      thebody.style.marginTop = "105px";
      if (thebody.style.width.lastIndexOf("px") == -1)
        thebody.style.width = "100%";
    }
  }
  enforceBanner();
</script>

<div id="webArchiveLogo" style="display: block; width: 317px; height: 70px; float:
left; margin: 15px 0px 5px 0px; background: url(/media/img/TNA_logo_white_201006.
gif) no-repeat 2px 20px;">&nbsp;</div>

<!-- the text content -->
<div style="display: block; margin-top: 18px; height: auto; width: 915px; margin-left:
325px; margin-right:0px; color: white !important; font-family: Verdana, Arial,
Helvetica, sans-serif; font-size: 12px; line-height: 14px; text-align: left; color:
white;">
...

```

Listing 11: HTML Snippet from PRONI Web Archive URI <http://webarchive.proni.gov.uk/20111214024729/http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?%3Furi=CELEX:32008L0056:EN:NOT>, showing the beginning of the PRONIBANNER section

```
...
<div id="PRONIBANNER" style="display: block; position: absolute; width: 100%; height: 96px
; margin: 0; padding: 0; top: 0; left: 0; z-index: 10; border: none; color: #18417f;
background-color: #ffffff; border-bottom: 4px solid #18417f;">
<div id="PRONILOGO" style="display: block; width: 400px; height: 96px; float: right;
margin: 0px; padding: 0; background-image: url('/media/img/pronilogo.jpg');
background-repeat: no-repeat; background-position: right bottom;"><span class="
ACCESSIBLE" style="visibility: hidden;">Public Record Office of Northern Ireland</
span>
</div>
<div id="PRONIBANNERCONTENT" style="display: block; height: 100%; padding: 0px 0px 0px
20px; font-family: Helvetica, Arial, sans-serif !important; font-size: 10.5pt !
important; text-align: left !important; line-height: 1.4em !important; overflow:
hidden;">
<div style="display: block; padding: 15px 0 2px 0; margin: 0; color: #18417f !
important; font-family: Helvetica, Arial, sans-serif !important; font-size: 10
pt !important; line-height: 1.40em !important; font-weight: bold !important;
background-color: transparent !important">THIS IS NOT A LIVE WEBSITE
...

```

Listing 12: HTML Snippet from Archive.is URI <http://archive.is/19961226114737/http://www.rsinc.com/>, showing the beginning of the HEADER section

```
...
<div id="HEADER" style="font-family:Verdana,Arial,Helvetica;background-color:#FFFAE1;
border-bottom:2px #B40010 solid;min-width:1028px"><div style="padding-top:10px"></div>
table style="width:1028px;font-size:10px" border="0" cellspacing="0" cellpadding="0">


```

Listing 13: HTML Snippet from Archive.is URI <http://archive.is/19961226114737/http://www.rsinc.com/>, showing the hashtags section allowing a user to quickly scroll through a memento

```
...
<table id="hashtags" style="text-align:right;font-family:Verdana,Arial,Helvetica;font-size
:10px" border="0" height="100%"><tr><td id="0%" style="vertical-align:top"><a style="
color:#999999" href="#0%">0%</a></td></tr><tr><td id="5%" style="vertical-align:top"><a
style="color:#999999" href="#5%">5%</a></td></tr><tr><td id="10%" style="vertical-align
:top"><a style="color:#999999" href="#10%">10%</a></td></tr><tr><td id="15%" style="
vertical-align:top"><a style="color:#999999" href="#15%">15%</a></td></tr><tr><td id="20%
" style="vertical-align:top"><a style="color:#999999" href="#20%">20%</a></td></tr><tr>
<td id="25%" style="vertical-align:top"><a style="color:#999999" href="#25%">25%</a></td>
</tr><tr><td id="30%" style="vertical-align:top"><a style="color:#999999" href="#30%">
30%</a></td></tr><tr><td id="35%" style="vertical-align:top"><a style="color:#999999"
href="#35%">35%</a></td></tr><tr><td id="40%" style="vertical-align:top"><a style="color
:#999999" href="#40%">40%</a></td></tr><tr><td id="45%" style="vertical-align:top"><a
style="color:#999999" href="#45%">45%</a></td></tr><tr><td id="50%" style="vertical-align
:top"><a style="color:#999999" href="#50%">50%</a></td></tr><tr><td id="55%" style="
vertical-align:top"><a style="color:#999999" href="#55%">55%</a></td></tr><tr><td id="60%
" style="vertical-align:top"><a style="color:#999999" href="#60%">60%</a></td></tr><tr>
<td id="65%" style="vertical-align:top"><a style="color:#999999" href="#65%">65%</a></td>
</tr><tr><td id="70%" style="vertical-align:top"><a style="color:#999999" href="#70%">
70%</a></td></tr><tr><td id="75%" style="vertical-align:top"><a style="color:#999999"
href="#75%">75%</a></td></tr><tr><td id="80%" style="vertical-align:top"><a style="color
:#999999" href="#80%">80%</a></td></tr><tr><td id="85%" style="vertical-align:top"><a
style="color:#999999" href="#85%">85%</a></td></tr><tr><td id="90%" style="vertical-align
:top"><a style="color:#999999" href="#90%">90%</a></td></tr><tr><td id="95%" style="
vertical-align:top"><a style="color:#999999" href="#95%">95%</a></td></tr><tr><td id="
100%" style="vertical-align:bottom;height:12px"><a style="color:#999999" href="#100%">
100%</a></td></tr></table>
...

```